



US009454548B1

(12) **United States Patent**  
**Chang et al.**

(10) **Patent No.:** **US 9,454,548 B1**  
(45) **Date of Patent:** **Sep. 27, 2016**

- (54) **PLUGGABLE STORAGE SYSTEM FOR DISTRIBUTED FILE SYSTEMS**
- (71) Applicant: **EMC Corporation**, Hopkinton, MA (US)
- (72) Inventors: **Lei Chang**, Beijing (CN); **Tao Ma**, Beijing (CN); **Zhanwei Wang**, Beijing (CN); **Lirong Jian**, Beijing (CN); **Lili Ma**, Beijing (CN); **Gavin Sherry**, San Mateo, CA (US)
- (73) Assignee: **EMC Corporation**, Hopkinton, MA (US)
- (\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 37 days.
- (21) Appl. No.: **13/843,067**
- (22) Filed: **Mar. 15, 2013**

**Related U.S. Application Data**

- (60) Provisional application No. 61/769,043, filed on Feb. 25, 2013.
- (51) **Int. Cl.**  
**G06F 7/00** (2006.01)  
**G06F 17/30** (2006.01)
- (52) **U.S. Cl.**  
CPC ..... **G06F 17/30283** (2013.01)
- (58) **Field of Classification Search**  
None  
See application file for complete search history.

**References Cited**

**U.S. PATENT DOCUMENTS**

5,655,116 A 8/1997 Kirk et al.  
5,706,514 A \* 1/1998 Bonola ..... G06F 9/3877 712/30  
6,266,682 B1 7/2001 LaMarca et al.

6,718,372 B1 \* 4/2004 Bober ..... G06F 17/30203 707/E17.032  
6,745,385 B1 \* 6/2004 Lupu ..... G06F 9/44521 717/148  
6,996,582 B2 \* 2/2006 Daniels ..... G06F 3/0607 707/781  
7,035,931 B1 4/2006 Zayas et al.  
7,069,421 B1 \* 6/2006 Yates, Jr. .... G06F 9/45533 712/209  
7,313,512 B1 \* 12/2007 Traut ..... G06Q 10/0631 703/23  
7,613,947 B1 11/2009 Coatney et al.  
7,689,535 B2 3/2010 Bernard  
7,702,625 B2 4/2010 Peterson et al.  
7,716,261 B2 \* 5/2010 Black ..... 707/827  
7,720,841 B2 5/2010 Gu et al.  
7,739,316 B2 \* 6/2010 Thompson ..... G06F 17/30595 707/803  
7,827,201 B1 \* 11/2010 Gordon ..... G06F 17/30607 707/792  
7,949,693 B1 \* 5/2011 Mason ..... G06F 21/6227 707/828  
7,958,303 B2 6/2011 Shuster  
7,978,544 B2 7/2011 Bernard  
8,028,290 B2 \* 9/2011 Rymarczyk ..... G06F 9/30181 712/16  
8,051,113 B1 \* 11/2011 Shekar ..... G06F 17/30067 707/821  
8,131,739 B2 \* 3/2012 Wu ..... G06F 17/30067 707/758  
8,180,813 B1 \* 5/2012 Goodson et al. .... 707/827  
8,185,488 B2 \* 5/2012 Chakravarty et al. .... 706/47

(Continued)

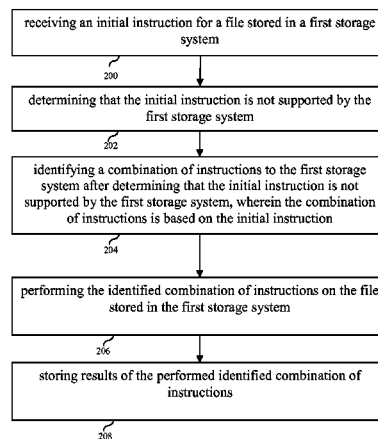
Primary Examiner — Hexing Liu

(74) *Attorney, Agent, or Firm* — Van Pelt, Yi & James LLP

(57) **ABSTRACT**

A method, article of manufacture, and apparatus for managing data. In some embodiments, this includes an initial instruction for a file stored in a first storage system, determining that the initial instruction is not supported by the first storage system, identifying a combination of instructions to the first storage system after determining that the initial instruction is not supported by the first storage system, wherein the combination of instructions is based on the initial instruction, wherein the combination of instructions is based on the initial instruction, performing the identified combination of instructions on the file stored in the first storage system, and storing results of the performed identified combination of instructions.

**17 Claims, 2 Drawing Sheets**



(56)

References Cited

U.S. PATENT DOCUMENTS

8,195,769 B2 6/2012 Miloushev et al.  
8,200,723 B1\* 6/2012 Sears ..... 707/828  
8,219,681 B1\* 7/2012 Glade ..... G06F 3/06  
707/616  
8,301,822 B2 10/2012 Pinto et al.  
8,312,037 B1 11/2012 Batchavachalu et al.  
8,352,429 B1 1/2013 Mamidi et al.  
8,417,681 B1 4/2013 Miloushev et al.  
8,452,821 B2 5/2013 Shankar et al.  
8,484,259 B1 7/2013 Makkar et al.  
8,533,183 B2 9/2013 Hokanson  
8,577,911 B1 11/2013 Stepinski et al.  
8,578,096 B2 11/2013 Malige et al.  
8,595,237 B1 11/2013 Chaudhary et al.  
8,682,853 B2 3/2014 Zane et al.  
8,682,922 B2 3/2014 Boneti  
8,762,330 B1 6/2014 Kick  
8,832,154 B1 9/2014 Srinivasan et al.  
8,971,916 B1 3/2015 Joyce et al.  
9,323,758 B1 4/2016 Stacey et al.  
2002/0002638 A1 1/2002 Obara  
2002/0049782 A1 4/2002 Herzenberg et al.  
2002/0133810 A1\* 9/2002 Giles ..... G06F 9/45533  
717/138  
2002/0146035 A1\* 10/2002 Tyndall ..... 370/465  
2003/0172094 A1 9/2003 Lauria et al.  
2003/0229637 A1 12/2003 Baxter et al.  
2004/0054748 A1 3/2004 Ackaouy et al.  
2004/0088282 A1 5/2004 Xu et al.  
2004/0098415 A1 5/2004 Bone et al.  
2004/0143571 A1 7/2004 Bjornson et al.  
2005/0165777 A1 7/2005 Hurst-Hiller et al.  
2005/0198401 A1 9/2005 Chron et al.  
2005/0216788 A1 9/2005 Mani-Meitav et al.  
2006/0005188 A1\* 1/2006 Vega ..... G06F 9/5077  
718/1  
2006/0010433 A1\* 1/2006 Neil ..... G06F 9/45533  
717/138  
2006/0037069 A1 2/2006 Fisher et al.  
2006/0136653 A1\* 6/2006 Traut ..... G06F 9/4406  
711/6  
2006/0146057 A1\* 7/2006 Blythe ..... G06F 9/45537  
345/506  
2006/0149793 A1 7/2006 Kushwah et al.  
2006/0173751 A1 8/2006 Schwarze et al.  
2006/0248528 A1\* 11/2006 Oney ..... G06F 9/45533  
718/1  
2008/0059746 A1 3/2008 Fisher

2008/0281802 A1 11/2008 Peterson et al.  
2008/0313183 A1 12/2008 Cunningham et al.  
2008/0320151 A1 12/2008 McCanne et al.  
2009/0007105 A1\* 1/2009 Fries ..... G06F 8/65  
718/1  
2009/0089344 A1 4/2009 Brown et al.  
2009/0106255 A1 4/2009 Lacapra et al.  
2009/0132609 A1 5/2009 Barsness et al.  
2009/0222569 A1 9/2009 Frick  
2009/0265400 A1\* 10/2009 Pudipeddi et al. .... 707/205  
2009/0328225 A1\* 12/2009 Chambers ..... G06F 21/10  
726/26  
2010/0036840 A1 2/2010 Pitts  
2010/0042655 A1 2/2010 Tse et al.  
2010/0145917 A1 6/2010 Bone et al.  
2010/0241673 A1\* 9/2010 Wu ..... G06F 17/30233  
707/812  
2010/0287170 A1 11/2010 Liu et al.  
2011/0113052 A1 5/2011 Hörnkqvist et al.  
2011/0137966 A1\* 6/2011 Srinivasan ..... H04L 67/1097  
707/828  
2011/0153662 A1 6/2011 Stanfill et al.  
2011/0153697 A1 6/2011 Nickolov et al.  
2011/0313973 A1\* 12/2011 Srivas et al. .... 707/634  
2012/0023145 A1 1/2012 Brannon et al.  
2012/0036107 A1 2/2012 Miloushev et al.  
2012/0066274 A1\* 3/2012 Stephenson ..... G06F 9/45533  
707/825  
2012/0095952 A1 4/2012 Archambeau et al.  
2012/0095992 A1 4/2012 Cutting et al.  
2012/0185913 A1 7/2012 Martinez et al.  
2012/0310916 A1 12/2012 Abadi et al.  
2012/0311572 A1\* 12/2012 Falls ..... G06F 9/455  
718/1  
2012/0317388 A1\* 12/2012 Driever et al. .... 711/170  
2013/0036272 A1 2/2013 Nelson  
2013/0166543 A1 6/2013 MacDonald et al.  
2013/0185735 A1\* 7/2013 Farrell ..... G06F 9/542  
719/318  
2013/0198716 A1\* 8/2013 Huang ..... G06F 9/44505  
717/121  
2013/0246347 A1\* 9/2013 Sorenson ..... G06F 17/30115  
707/609  
2013/0275653 A1 10/2013 Ranade et al.  
2014/0136483 A1 5/2014 Chaudhary et al.  
2014/0149392 A1 5/2014 Wang et al.  
2014/0188845 A1 7/2014 Ah-Soon et al.  
2014/0337323 A1 11/2014 Soep et al.  
2015/0120711 A1 4/2015 Liensberger et al.

\* cited by examiner

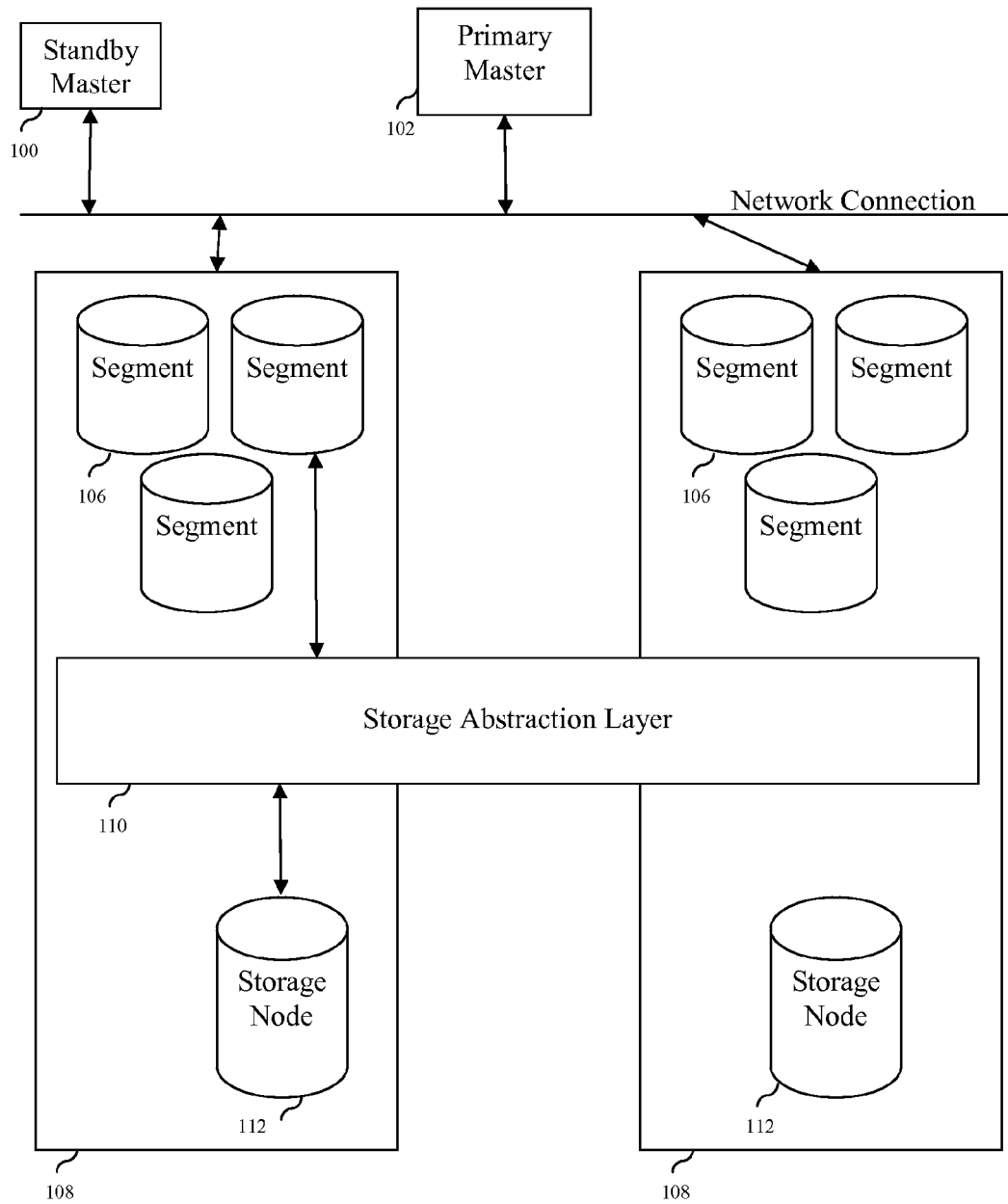


FIG. 1

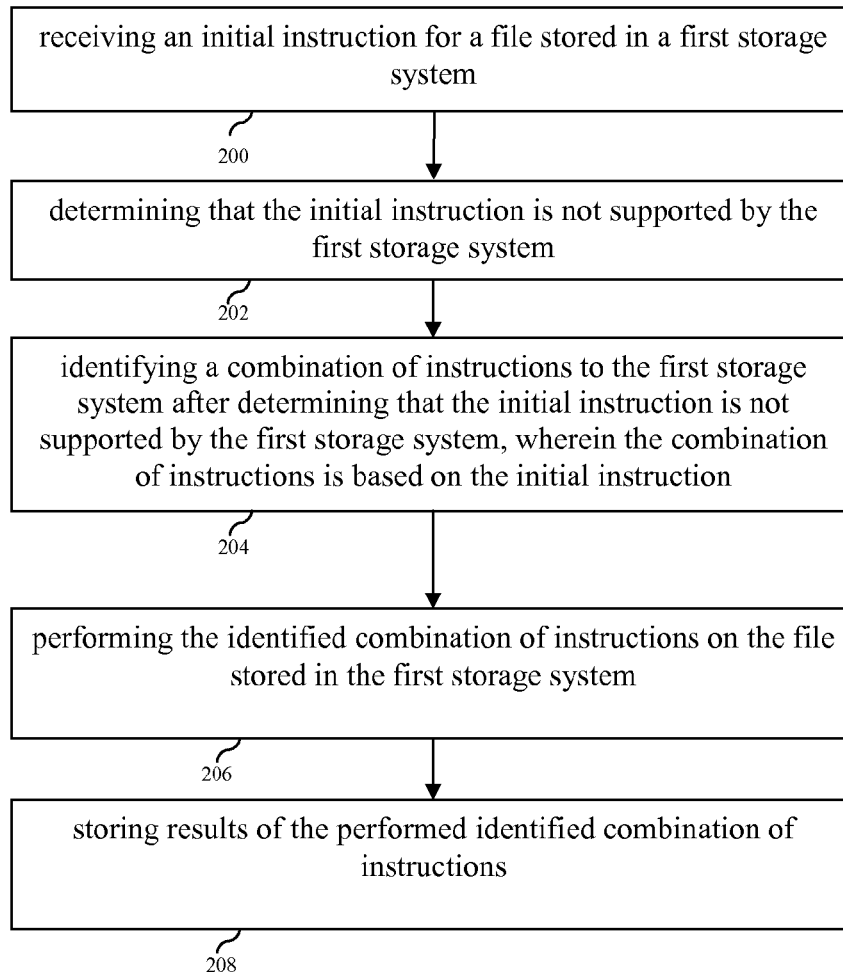


FIG. 2

1

## PLUGGABLE STORAGE SYSTEM FOR DISTRIBUTED FILE SYSTEMS

### CROSS REFERENCE TO RELATED APPLICATIONS

This application claims priority to U.S. Provisional Patent Application No. 61/769,043 for INTEGRATION OF MASSIVELY PARALLEL PROCESSING WITH A DATA INTENSIVE SOFTWARE FRAMEWORK filed on Feb. 25, 2013, which is incorporated herein by reference for all purposes.

### FIELD OF THE INVENTION

This invention relates generally to databases, and more particularly to systems and methods for managing datasets in databases.

### BACKGROUND OF THE INVENTION

With the large amounts of data generated in recent years, data mining and machine learning are playing an increasingly important role in today's computing environment. For example, businesses may utilize either data mining or machine learning to predict the behavior of users. This predicted behavior may then be used by businesses to determine which plan to proceed with, or how to grow the business.

The data used in data mining and analytics is typically not stored in a uniform data storage system. Many data storage systems utilize different file systems, and those different file systems are typically not compatible with each other. Further, the data may reside in geographically diverse locations.

One conventional method to performing data analytics across different databases includes copying data from one database to a central database, and performing the data analytics on the central database. However, this results in an inefficient use of storage space, and creates issues with data consistency between the two databases.

There is a need, therefore, for an improved method, article of manufacture, and apparatus for managing data.

### BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be readily understood by the following detailed description in conjunction with the accompanying drawings, wherein like reference numerals designate like structural elements, and in which:

FIG. 1 illustrates a database system in accordance with some embodiments.

FIG. 2 is a flowchart of a method to manage data in accordance with some embodiments.

### DETAILED DESCRIPTION

A detailed description of one or more embodiments of the invention is provided below along with accompanying figures that illustrate the principles of the invention. While the invention is described in conjunction with such embodiment(s), it should be understood that the invention is not limited to any one embodiment. On the contrary, the scope of the invention is limited only by the claims and the invention encompasses numerous alternatives, modifications, and equivalents. For the purpose of example, numerous specific details are set forth in the following description in order to provide a thorough understanding of the present

2

invention. These details are provided for the purpose of example, and the present invention may be practiced according to the claims without some or all of these specific details. For the purpose of clarity, technical material that is known in the technical fields related to the invention has not been described in detail so that the present invention is not unnecessarily obscured.

It should be appreciated that the present invention can be implemented in numerous ways, including as a process, an apparatus, a system, a device, a method, or a computer readable medium such as a computer readable storage medium or a computer network wherein computer program instructions are sent over optical or electronic communication links. Applications may take the form of software executing on a general purpose computer or be hardwired or hard coded in hardware. In this specification, these implementations, or any other form that the invention may take, may be referred to as techniques. In general, the order of the steps of disclosed processes may be altered within the scope of the invention.

An embodiment of the invention will be described with reference to a data storage system in the form of a storage system configured to store files, but it should be understood that the principles of the invention are not limited to this configuration. Rather, they are applicable to any system capable of storing and handling various types of objects, in analog, digital, or other form. Although terms such as document, file, object, etc. may be used by way of example, the principles of the invention are not limited to any particular form of representing and storing data or other information; rather, they are equally applicable to any object capable of representing information.

FIG. 1 illustrates a database system in accordance with some embodiments. Primary Master **102** accepts queries, plans queries, and dispatches queries to Segments **106** for execution. Primary Master **102** also collects results from Segments **106**. Segments **106** are each a compute unit of the database system. Rack **108** can store multiple segments as hosts (hosts not shown in FIG. 1). Standby Master **100** is a warm backup of the primary host. Primary Master **102** and Standby Master **100** communicate to Rack **108** via the network connection. The tables of the database system are stored in Storage Nodes **112**. Access, including read and write, are done through Storage Abstraction Layer **110**. Primary Master **102**, in some embodiments, may include a meta store (not shown in FIG. 1).

The meta store includes information about the different files systems in Storage Nodes **112**, such as API information of the file system's interface, and different attributes and metadata of the file system. The meta store also includes information on the binary location of the Storage Abstraction Layer **110**. As new file systems are added to the database system, the new file systems are registered (e.g. provided API information, other attributes, etc.) with the meta store. Once the new file systems are added, instances of that file system may be created to store data objects, such as databases and tables.

The storage nodes may be different file systems. For example, one storage node may be Hadoop File System (HDFS), while another storage node may be NFS. Having multiple file systems presents some challenges. One challenge is that file systems do not support all the same commands. The Storage Abstraction Layer helps address this challenge.

In some embodiments, the Storage Abstraction Layer selects a file system instance. A file system instance means a physical storage system for a specific file system. As

discussed above, there may be several different file systems, and several different instances. The instances may be of the same file system, or they may be of different file systems. Different file systems may have different semantics or different performance characteristics. For example, some file systems allow you to update data, while other file systems only let you append data. The Storage Abstraction Layer chooses a file system based on the file system's attributes.

For example, in some embodiments, if a user wanted to modify or update a file that is stored on an underlying storage system which does not support file modification, the Storage Abstraction Layer may recognize the update command and move the file from the underlying storage system to another which does support file modification. The move may be temporary (e.g. move the file back after the user is finished with the file), or the move may be permanent.

In some embodiments, the Storage Abstraction Layer may choose to store a data object in a file system that does not allow updating. This may be preferable in cases where the data object is only read and never modified, and the file system is efficient for retrieving data. Thus, the Storage Abstraction Layer may take into account the usage statistics of the data object to determine what file system to use to store the data.

In some embodiments, the Storage Abstraction Layer may perform semantic adaptation. This may be preferable when the underlying file system may not be able to communicate directly with segments. This may occur when the interface the Storage Abstraction Layer exposes to the segment execution engine does not match with the semantics of the underlying file system. Other examples include instances where the functionality required by the segments is not supported by the underlying file system.

For example, a user may wish to truncate a file. However, the file may be stored on a segment where the underlying storage does not allow truncating files. The user is not aware of this because the user is not aware of where the files are physically stored. Typically, without a Storage Abstraction Layer, the underlying file system would not be able to understand the truncate command.

One example of semantic adaption includes adapting the truncate command. Suppose that a segment requires a piece of data to be truncated. However, the underlying file system does not support the truncate functionality. The Storage Abstraction Layer may be able to put various commands together to mimic a truncate command. Since the Storage Abstraction Layer has access to the metadata of the file system stored in the meta store, it knows what commands are allowed in the file system, as well as how to access the file system via APIs. Suppose that the file to be truncated is File A, and File A consists of 20 bytes. The segment wants the last 10 bytes to be deleted. With this requirement, the Storage Abstraction Layer may employ semantic adaptation to complete the truncation even when the underlying file system does not support a truncate command. In some embodiments, the Storage Abstraction Layer may first copy the first 10 bytes of File A to a temporary file, called File B. Then, the original File A is deleted, leaving only the temporary File B. After the original File A is deleted, the temporary File B is renamed to File A. File A is now only half of the original File A. In other words, File A has been truncated, even though the underlying file system did not support truncation. The Storage Abstraction Layer, by understanding how to access the underlying file system via the meta store, sent a series of commands to mimic a truncate. This series of commands may be stored in the meta store so that future truncate requests may make use of it.

Another example of semantic adaption includes a file update command. As mentioned above, some file systems do not allow for updating a file. Suppose a segment required that a file be updated. However, the file is stored in a file system that does not allow files to be updated. In some embodiments, the Storage Abstraction Layer may record the modifications in separate file as a new version. For example, if File A was to be modified, the separate file may be called File A\_ver2. The segment (or user) will see that changes are being made to File A, but in fact, File A remains unchanged and the changes are being stored in File A\_ver2. After the segment is finished modifying or updating the file, there may be two files stored—one is File A, and the other is File A\_ver2. When subsequent users want to access File A, the Storage Abstraction Layer may cause the two files to be merged. With File A merged with File A\_ver2 and called File A, the new File A will include all the changes made by the previous user. In other words, File A has been modified, even though the underlying file system did not support updating.

With the Storage Abstraction Layer, many different file systems may be supported. New and different storage systems with different file systems may be "plugged" into the database, without affecting the ability for the database to run its queries or jobs, as long as the meta store is updated with information about the new file system, such as its APIs.

FIG. 2 illustrates a method to manage data in accordance with some embodiments. In step 200, an initial instruction for a file stored in a first storage system is received. In step 202, it is determined that the initial instruction is not supported by the first storage system. In step 204, a combination of instructions to the first storage system is identified after determining that the initial instruction is not supported by the first storage system, wherein the combination of instructions is based on the initial instruction. In step 206, the identified combination of instructions is performed on the file stored in the first storage system. In step 208, results of the performed identified combination of instructions are stored.

For the sake of clarity, the processes and methods herein have been illustrated with a specific flow, but it should be understood that other sequences may be possible and that some may be performed in parallel, without departing from the spirit of the invention. Further, though the techniques herein teach creating one SwR sample in parallel, those with ordinary skill in the art will readily appreciate that the techniques are easily extendable to generate many SwR samples. Additionally, steps may be subdivided or combined. As disclosed herein, software written in accordance with the present invention may be stored in some form of computer-readable medium, such as memory or CD-ROM, or transmitted over a network, and executed by a processor.

All references cited herein are intended to be incorporated by reference. Although the present invention has been described above in terms of specific embodiments, it is anticipated that alterations and modifications to this invention will no doubt become apparent to those skilled in the art and may be practiced within the scope and equivalents of the appended claims. More than one computer may be used, such as by using multiple computers in a parallel or load-sharing arrangement or distributing tasks across multiple computers such that, as a whole, they perform the functions of the components identified herein; i.e. they take the place of a single computer. Various functions described above may be performed by a single process or groups of processes, on a single computer or distributed over several computers. Processes may invoke other processes to handle certain tasks. A single storage device may be used, or several may

5

be used to take the place of a single storage device. The disclosed embodiments are illustrative and not restrictive, and the invention is not to be limited to the details given herein. There are many alternative ways of implementing the invention. It is therefore intended that the disclosure and following claims be interpreted as covering all such alterations and modifications as fall within the true spirit and scope of the invention.

What is claimed is:

1. A method for managing data, comprising:
  - storing a file in a first storage system having a first file system, wherein the first storage system is selected by a storage abstraction layer based at least in part on whether file system functionality associated with usage statistics of the file is supported by the first file system of the first storage system;
  - receiving an initial instruction associated with performing a first action in relation to a file stored in a first storage system;
  - determining that the initial instruction associated with performing the first action is not supported by the first storage system based at least in part on metadata associated with the first storage system, wherein the metadata associated with the first storage system is stored in meta store that includes information respectively associated with one or more storage systems;
  - in response to determining that the initial instruction associated with performing the first action is not supported by the first storage system, identifying a combination of instructions to the first storage system, wherein the identifying of the combination of instructions includes storing an update in a second file and merging the second file with the file after the update is complete, wherein the combination of instructions is based on the initial instruction and performs the first action by performing a series of actions that have a collective result that is equivalent to a result of the first action, and wherein the combination of instructions are determined based at least in part on the metadata associated with the first storage system, and wherein the metadata associated with the first storage system indicates a mechanism for accessing a file system corresponding to the first storage system;
  - performing the identified combination of instructions on the file stored in the first storage system; and
  - storing results of the performed identified combination of instructions.
2. The method as recited in claim 1, wherein the initial instruction includes a truncate instruction.
3. The method as recited in claim 2, wherein the combination of instructions includes copy a first portion of the file, delete the file, and rename the first portion of the file.
4. The method as recited in claim 1, wherein the initial instruction includes an update instruction.
5. A system for managing data, comprising a storage device and a processor configured to:
  - store a file in a first storage system having a first file system, wherein the first storage system is selected by a storage abstraction layer based at least in part on whether file system functionality associated with usage statistics of the file is supported by the first file system of the first storage system
  - receive an initial instruction associated with performing a first action in relation to a file stored in a first storage system;
  - determine that the initial instruction associated with performing the first action is not supported by the first

6

- storage system based at least in part on metadata associated with the first storage system, wherein the metadata associated with the first storage system is stored in meta store that includes information respectively associated with one or more storage systems;
- in response to determining that the initial instruction associated with performing the first action is not supported by the first storage system, identify a combination of instructions to the first storage system, wherein to identify the combination of instructions includes to store an update in a second file and merging the second file with the file after the update is complete, wherein the combination of instructions is based on the initial instruction and performs the first action by performing a series of actions that have a collective result that is equivalent to a result of the first action, and wherein the combination of instructions are determined based at least in part on the metadata associated with the first storage system, and wherein the metadata associated with the first storage system indicates a mechanism for accessing a file system corresponding to the first storage system;
- perform the identified combination of instructions on the file stored in the first storage system; and
- store results of the performed identified combination of instructions.
6. The system as recited in claim 5, wherein the initial instruction includes a truncate instruction.
7. The system as recited in claim 6, wherein the combination of instructions includes copy a first portion of the file, delete the file, and rename the first portion of the file.
8. The system as recited in claim 5, wherein the initial instruction includes an update instruction.
9. A computer program product for processing data, comprising a non-transitory computer readable medium having program instructions embodied therein for:
  - storing a file in a first storage system having a first file system, wherein the first storage system is selected by a storage abstraction layer based at least in part on whether file system functionality associated with usage statistics of the file is supported by the first file system of the first storage system;
  - receiving an initial instruction associated with performing a first action in relation to a file stored in a first storage system;
  - determining that the initial instruction associated with performing the first action is not supported by the first storage system based at least in part on metadata associated with the first storage system, wherein the metadata associated with the first storage system is stored in meta store that includes information respectively associated with one or more storage systems;
  - in response to determining that the initial instruction associated with performing the first action is not supported by the first storage system, identifying a combination of instructions to the first storage system, wherein the identifying of the combination of instructions includes storing an update in a second file and merging the second file with the file after the update is complete, wherein the combination of instructions is based on the initial instruction and performs the first action by performing a series of actions that have a collective result that is equivalent to a result of the first action, and wherein the combination of instructions are determined based at least in part on the metadata associated with the first storage system, and wherein the metadata associated with the first storage system

7

indicates a mechanism for accessing a file system corresponding to the first storage system; performing the identified combination of instructions on the file stored in the first storage system; and storing results of the performed identified combination of instructions.

**10.** The computer program product as recited in claim **9**, wherein the initial instruction includes a truncate instruction.

**11.** The computer program product as recited in claim **10**, wherein the combination of instructions includes copy a first portion of the file, delete the file, and rename the first portion of the file.

**12.** The computer program product as recited in claim **9**, wherein the initial instruction includes an update instruction.

**13.** The method of claim **1**, wherein the determining that the initial instruction associated with the first action is not supported by the first storage system and the identifying of the combination of instructions are performed, or caused to be performed, by the storage abstraction layer.

8

**14.** The method of claim **13**, wherein the storage abstraction layer selects a file system of the one or more storage systems in which the file is to be stored based at least in part on a usage statistic of the file and the metadata associated with the corresponding file storage system.

**15.** The method of claim **13**, wherein in response to a second storage system being added, the storage abstraction layer updates the meta store to include information corresponding to the second storage system.

**16.** The method of claim **1**, wherein the information respectively associated with the one or more storage systems comprises one or more of an Application Program Interface (API) information of a corresponding interface of the one or more storage systems, an attribute of the one or more storage systems, and metadata associated with the one or more storage systems.

**17.** The method of claim **1**, wherein the combination of instructions comprises a plurality of actions that are collectively mapped to the initial instruction.

\* \* \* \* \*